Roteiro de atividade prática

Nome: Turma: .

**Título da atividade: Melhorando a implementação**

**Objetivos**

Os alunos devem analisar o código da aula anterior e melhorar conforme instruções.

**Lista de materiais**

* Um computador com *browser* de internet;
* Caderno para anotações;
* 1 caneta.

**Procedimento experimental**

1. Acesse o site Colab do Google e crie um notebook.
2. Copie o código da aula no bloco de notas e cole-o no notebook.

import numpy as np

import tensorflow as tf

# Definir o tamanho do batch

batch\_size = 16

# Definir o tamanho da sequência de notas

seq\_len = 128

# Definir o número de notas

n\_notes = 128

# Definir a função de perda

def loss\_fn(real, fake):

return tf.reduce\_mean(tf.math.square(real - fake))

# Definir o gerador

class Generator(tf.keras.Model):

def \_\_init\_\_(self):

super(Generator, self).\_\_init\_\_()

self.dense1 = tf.keras.layers.Dense(128, activation='relu')

self.dense2 = tf.keras.layers.Dense(128, activation='relu')

self.dense3 = tf.keras.layers.Dense(n\_notes, activation='tanh')

def call(self, inputs):

x = inputs

x = self.dense1(x)

x = self.dense2(x)

x = self.dense3(x)

return x

# Definir o discriminador

class Discriminator(tf.keras.Model):

def \_\_init\_\_(self):

super(Discriminator, self).\_\_init\_\_()

self.dense1 = tf.keras.layers.Dense(128, activation='relu')

self.dense2 = tf.keras.layers.Dense(128, activation='relu')

self.dense3 = tf.keras.layers.Dense(1, activation='sigmoid')

def call(self, inputs):

x = inputs

x = self.dense1(x)

x = self.dense2(x)

x = self.dense3(x)

return x

# Instanciar os modelos

generator = Generator()

discriminator = Discriminator()

# Compilar os modelos

generator.compile(loss=loss\_fn, optimizer='adam')

discriminator.compile(loss='binary\_crossentropy', optimizer='adam')

# Criar o loop de treinamento

for epoch in range(100):

# Gerar um batch de notas aleatórias

real\_notes = np.random.randint(0, n\_notes, (batch\_size, seq\_len))

# Passar as notas aleatórias pelo discriminador

real\_output = discriminator(real\_notes)

# Gerar um batch de notas sintéticas

fake\_notes = generator(np.random.randint(0, 1, (batch\_size, seq\_len)))

# Passar as notas sintéticas pelo discriminador

fake\_output = discriminator(fake\_notes)

# Atualizar o discriminador

discriminator.train\_on\_batch(real\_notes, np.ones((batch\_size, 1)))

discriminator.train\_on\_batch(fake\_notes, np.zeros((batch\_size, 1)))

# Atualizar o gerador

generator.train\_on\_batch(np.random.randint(0, 1, (batch\_size, seq\_len)), np.ones((batch\_size, 1)))

# Gerar notas sintéticas

generated\_notes = generator(np.random.randint(0, 1, (1, seq\_len)))

# Mostrar as notas sintéticas

print(generated\_notes)

1. Clique em “Play” para executar. Inicialmente, vamos melhorar o código anterior verificando e inserindo no código as bibliotecas necessárias.
2. Insira ao final do código as linhas para:

|  |
| --- |
| audio = librosa.core.midi\_to\_hz(generated\_notes.reshape(max\_seq\_len) )  # Salvar o audio  ipd.Audio(audio, rate=44100). |

1. Experimente diferentes configurações do modelo. Por exemplo, você pode aumentar o tamanho do *batch*, o número de notas, para gerar áudio mais complexo, o número de treinamento.

|  |
| --- |
|  |
|  |
|  |